HMOE: Hypernetwork-based Mixture of Experts for Domain Generalization

Jingang Qu, Thibault Faney, Ze Wang, Patrick Gallinari, Soleiman Yousef, Jean-Charles de Hemptinne

HMOE: Hypernetwork-based Mixture of Experts for Domain Generalization

Jingang Qu[1,2] Thibault Faney[2] Ze Wang[1] Patrick Gallinari[1,3]
Soleiman Yousef[2] Jean-Charles de Hemptinne[2]

Sorbonne Université, CNRS, ISIR, F-75005 Paris, France[1] IFPEN [2] Criteo AI Lab, Paris [3]

Abstract

Due to the domain shift, machine learning systems typically fail to generalize well to domains different from those of training data, which is the problem that domain generalization (DG) aims to address. However, most mainstream DG algorithms lack interpretability and require domain labels, which are not available in many real-world scenarios. In this work, we propose a novel DG method, HMOE: Hypernetwork-based Mixture of Experts (MoE), that does not require domain labels and is more interpretable. We use hypernetworks to generate the weights of experts, allowing experts to share some useful meta-knowledge. MoE has proven adept at detecting and identifying heterogeneous patterns in data. For DG, heterogeneity exactly arises from the domain shift. We compare HMOE with other DG algorithms under a fair and unified benchmark-DomainBed. Extensive experiments show that HMOE can perform latent domain discovery from data of mixed domains and divide it into distinct clusters that are surprisingly more consistent with human intuition than original domain labels. Compared to other DG methods, HMOE shows competitive performance and achieves SOTA results in some cases without using domain labels.

1. Introduction

Domain generalization (DG) aims to train models on known domains that can generalize well to unseen domains, which is of crucial importance for deploying models in safety-critical real-world applications. Over the past decade, great efforts have been made to develop a variety of DG algorithms [26, 82, 90], most of which have focused on developing DG-specific data augmentation techniques and learning domain-invariant representations on which to build generalizable predictors. However, many high-performing DG algorithms entail the knowledge of domain labels to explicitly reduce domain discrepancy, which severely limits their applicability in real-world scenarios where domain annotation may be prohibitively expensive. In addition, current algorithms fall short of interpretability and cannot provide insight into the causes of success or failure in generalizing to new domains. Therefore, our work aims to propose a novel DG algorithm that does not require domain labels and has good interpretability.

We follow the nomenclature of [9], which refers to DG with domain labels as vanilla DG and the more challenging DG without domain labels as compound DG. It was shown in [5, 12, 54] that domain information plays an important role in obtaining better DG performance. Therefore, the key to solving compound DG is how to infer domain information from the data of mixed domains. To make the problem tractable, we assume that latent domains are distinct and separable.

In this work, we propose HMOE: Hypernetwork-based Mixture of Experts (MoE). MoE is a well-established learning paradigm that combines several experts by calculating the weighted sum of their predictions [33, 34], where the aggregation weights, also known as gate values, add up to 1 and are determined by a routing mechanism. An innovation of our work is to use a neural network, called a hypernetwork [27], to generate the weights of expert networks. In this way, the hypernetwork serves as a link between experts and provides a platform for them to exchange information.

Our work leverages MoE's *divide and conquer* property, that is, the routing mechanism learns to route inputs to different experts in an unsupervised manner and softly partitions the input space into subspaces [85], and each expert becomes specialized in a subspace. This property makes MoE a natural choice for discovering heterogeneous patterns in data. We further expect that each subspace is associated with a latent domain, thus enabling latent domain discovery. In addition, thanks to the probabilistic nature of MoE, HMOE can be easily extended to support semi-supervised learning on partial domain labels. During inference, when faced with an unseen test domain, we can compare the similarities between the test domain and the inferred domains based on gate values, hence improving interpretability.

However, MoE's intrinsic soft partitioning is not always effective and sometimes fails to maintain a consistent di-

vision of the input space, especially when the distinction between latent domains is not significant. Therefore, we propose a differentiable dense-to-sparse Top-1 routing algorithm, which forces gate values to become one-hot and converges to hard partitioning. In this way, we achieve sparse MoE and enhance and stabilize latent domain discovery. In addition, we devise a novel method for calculating gate values to better incorporate hypernetworks into MoE.

We summarize our contributions as follows: (1) We propose a principled and conceptually simple approach, HMOE, for compound DG, which can discover latent domains, has good interpretability, and is trained in an end–to-end manner. (2) To our best knowledge, we are the first to combine hypernetworks and MoE to solve the DG problem. (3) We undertake comprehensive experiments to compare HMOE with other DG methods under a fair and unified evaluation framework - DomainBed [26]. HMOE achieves competitive performance and even state-of-the-art results in some cases without requiring domain labels.

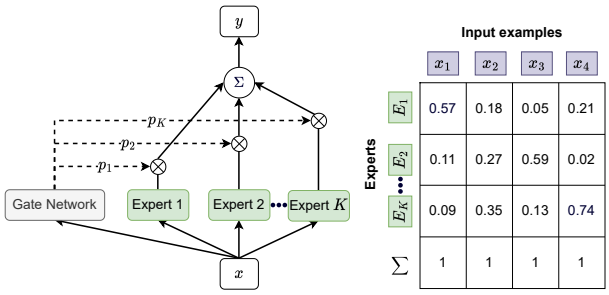
2. Related Work

2.1. Domain Generalization (DG)

The goal of DG is to train a predictor on known domains that can generalize well to unseen domains.

Vanilla DG The first line of work is to design DG-specific data augmentation techniques to increase the diversity and quantity of training data to improve DG performance [48, 61, 66, 79, 84, 86, 91, 93]. Previous work learned domain-invariant representations through invariant risk minimization [1, 2, 38], kernel methods [5, 20, 23, 54], feature alignment [24, 44, 52, 53, 57, 59, 70, 72, 81], and domain-adversarial training [21, 22, 25, 44, 46]. Another approach is to disentangle latent features into class-specific and domain-specific representations [32, 35, 55, 60, 87]. General machine learning paradigms were also applied to vanilla DG, such as meta-learning [3, 15, 41, 43], self-supervised learning [7, 36], gradient manipulation [31, 62, 68], and distributionally robust optimization [38, 63].

Compound DG There are some DG algorithms that do not require domain labels by design [9, 31, 45, 52, 55, 87]. In addition to improving DG performance, latent domain discovery is also an important task in compound DG and contributes to better interpretability. [9, 52] can do this but have two main limitations: (1) Their proposed methods proceed in two phases: first discover potential domains, and then deal with DG with the inferred domains, which is similar to vanilla DG. The problem is that the second phase depends on the first and cannot provide some feedback to correct possible errors in domain discovery. (2) Their methods assume that domain discrepancy arises from stylistic differences in order to identify latent domains, which does not always hold.



(a) Classical Mixture of Experts (b) Gate value matrix

Figure 1. (a) Mixture of Experts calculates the weighted sum of experts' outputs. (b) The aggregation weights, also known as gate values, are calculated by the gate network on a per-example basis.

In our work, all components are jointly optimized in an end-to-end fashion. In addition, we leverage MoE to find latent domains without an explicit induced bias on the cause of domain discrepancy.

2.2. Hypernetworks

A hypernetwork is a neural network that generates the weights of another target network. Hypernetworks were initially proposed by [27] and then applied to optimization problems [49, 56], meta-learning [89], continuous learning [6, 80], multi-task learning [47, 50, 71], few-shot learning [64], and federated learning [65].

2.3. Mixture of Experts (MoE)

Mixture of Experts (MoE), originally proposed by [33, 34], consists of two main components: experts and a gate network, as shown in Fig. 1a. Its output is a weighted sum of experts, and the gate network calculates gate values on a per-example basis, as shown in Fig. 1b. In the past few years, MoE has regained attention as a way to scale up deep learning models without significantly increasing computational cost and to more effectively harness modern hardware [16, 18, 19, 39, 67, 94]. In this case, sparse MoE is used, which routes each example only to the experts with Top-1 or Top-K gate values, instead of all of them.

2.4. Application of Hypernetworks and MoE in DG

To the best of our knowledge, no work has used hypernetworks to solve DG in the field of computer vision. Recently, [78] applied hypernetworks to DG in natural language processing (NLP) and achieved state-of-the-art results on two NLP-related DG tasks. As for MoE, [40] proposed replacing feed-forward network layer (FFN) of Vision Transformer (ViT) [14] with a sparse mixture of FFN experts to improve DG performance. In addition, if we regard MoE as a kind of ensemble method, there are some work having the same spirit [13, 51, 92].

3. Method

3.1. Problem Setting

Let $\mathcal{X}$ denote an input space and $\mathcal{Y}$ a target space. A domain S is characterized by a joint distribution P^s_{XY} on $\mathcal{X} \times \mathcal{Y}$. In vanilla DG setting, we have a training set containing M known domains, *i.e.*, $\mathcal{D}^V_{tr} = \{\mathcal{D}^s\}^M_{s=1}$ with $\mathcal{D}^s = \{(x^s_i, y^s_i, d^s_i)\}^{N_s}_{i=1}$ where $(x^s_i, y^s_i) \sim P^s_{XY}$ and d^s_i is the domain index or label. Also consider a test dataset $\mathcal{D}_{te}$ composed of unknown domains different from those of $\mathcal{D}^V_{tr}$. Vanilla DG aims to train a robust predictor $f : \mathcal{X} \to \mathcal{Y}$ on $\mathcal{D}^V_{tr}$ with domain labels to achieve a minimum predictive error on $\mathcal{D}_{te}$, *i.e.*, $\min_f \mathbb{E}_{(x,y)\sim\mathcal{D}_{te}}[\ell(f(x), y)]$, where $\ell(\cdot, \cdot)$ is the loss function.

Our work focuses on the more difficult compound DG, for which the training set $\mathcal{D}_{tr} = \{(x_i, y_i)\}^N_{i=1}$ contains mixed domains and therefore has no domain annotation. However, as demonstrated in [26, 82, 90], intrinsic inter-domain relationships play a key role in obtaining better generalization performance. Therefore, our proposed HMOE is required to identify and discover latent domains by dividing $\mathcal{D}_{tr}$ into clusters that match human intuition about visual relationships between different domains.

3.2. Overall Architecture

An overview of the HMOE architecture is depicted in Fig. 2a. It processes each input x through two paths: the domain path, which aims at performing latent domain discovery, and the classifier path, which aims at training a classifier expert for each latent domain. The classifier path starts with a featurizer h_z to extract high-level features from x, which can be a pretrained network, such as VGG [69], ResNet [29], and ViT. We define a discrete learnable embedding space $\mathcal{E}$ with K embedding vectors $\{e_k \in \mathbb{R}^D\}^K_{k=1}$, which are fed into a hypernetwork f_h to generate a set of weights $\{\theta_k\}^K_{k=1}$. These weights further form a set of classifiers $\{f_c(:;\theta_k)\}^K_{k=1}$. The output of the featurizer is passed to these K classifier experts to compute their corresponding outputs $y_k = f_c(z;\theta_k)$.

The domain path starts with a Domain2Vec (D2V) encoder h_v, which transforms x into the embedding space $\mathcal{E}$ and outputs $v \in \mathbb{R}^D$. The output v is then compared with the embedding vectors through a predefined gate function $g(v, \mathcal{E})$, as shown in Fig. 2b, to produce a set of probabilities $\boldsymbol{p} = \{p_k\}^K_{k=1}$. The final output y is the weighted sum of experts' outputs:

$$y = \sum_{k=1}^{K} p_k y_k = \langle g(h_v(x), \mathcal{E}), [f_c(h_z(x); f_h(e_k))]^K_{k=1} \rangle \tag{1}$$

In classical MoE, the gate network and experts have the same input. On the contrary, in our work, the D2V encoder takes images as input rather than the featurizer's extracted features, which mainly contain class-specific information for classification. If we link the D2V encoder to the featurizer, HMOE risks separating the input space based on semantic categories rather than domain-wise distinction.

3.3. Hypernetworks

We use the hypernetwork f_h taking as input a vector e to generate the weights of the classifier f_c. In our work, both f_h and f_c are MLPs. In a sense, f_c is just a placeholder computational graph, e can be viewed as a conditioning signal, and f_h maps e into a function. In contrast to classical MoE with a number of experts, we can use f_h to generate many experts without significantly increasing model parameters. In classical MoE, there is no direct communication between experts. In our work, experts are able to share some meta-knowledge through f_h. In addition, we use the hyperfan method proposed by [8] to initialize f_h.

3.4. Routing Mechanism

3.4.1 Gate Function

We need to calculate gate values $\boldsymbol{p}$ to quantify the responsibilities of experts for each input example and to aggregate experts' outputs. Based on the output of the D2V encoder v and the embedding space $\mathcal{E}$, we define a gate function $g(v, \mathcal{E})$ to calculate $\boldsymbol{p}$ as follows (Fig. 2b):

$$d_k = \|v - e_k\|_2 \tag{2a}$$

$$s_k = -\log(d^2_k + \epsilon) \tag{2b}$$

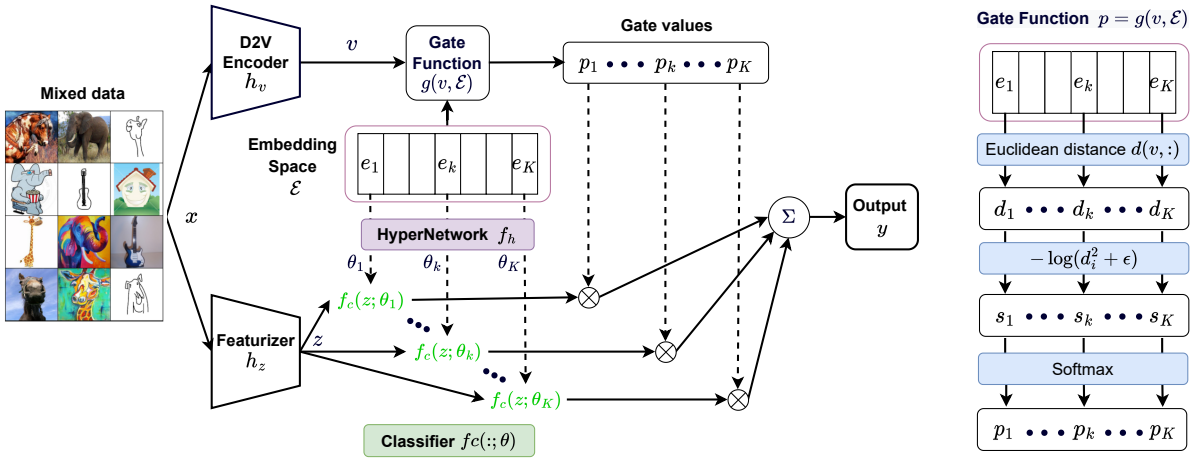
$$p_k = \frac{\exp(s_k)}{\sum^K_{j=1} \exp(s_j)} \tag{2c}$$

where ϵ is a small value. The negative logarithm in Eq. (2b) is used to establish a negative correlation between d_k and p_k (*i.e.*, the smaller d_k, the larger p_k) and to nonlinearly rescale the distance d (*i.e.*, stretch small d and squeeze great d), which makes $\boldsymbol{p}$ less sensitive to large d.

3.4.2 Differentiable Dense-to-Sparse Top-1 Routing

Based on gate values $\boldsymbol{p}$, the routing algorithm determines where and how to route input examples. A consistent and cohesive routing is essential for the training stability and convergence of MoE [10]. To enhance and stabilize latent domain discovery to capture less obvious domain differences, we would like to realize sparse MoE. However, the commonly used Top-1 or Top-K functions are not differentiable and cause oscillatory behavior of gate values during training [28]. Therefore, we propose a differentiable dense-to-sparse Top-1 routing algorithm by introducing an entropy loss on $\boldsymbol{p}$:

$$\mathcal{L}_{en} = \mathbb{E}_{(x,y)\sim\mathcal{D}_{tr}} [\mathbb{H}(g(h_v(x), \mathcal{E}))] \tag{3}$$



(a) Architecture (b) Gate function

Figure 2. (a) An overview of HMOE. In the upper branch, the input is transformed into the embedding space through the D2V encoder and gate values are calculated by a predefined gate function. In the lower branch, the hypernetwork takes as input embedding vectors to create a set of classifiers. The output is the weighted sum of classifiers' predictions. (b) The gate function calculates gate values based on the distances between the output of the D2V encoder and the embedding vectors. The smaller the distance, the greater the gate value.

where $\mathbb{H}(\cdot)$ denotes the entropy of a distribution. In practice, we multiply $\mathcal{L}_{en}$ by γ_{en} that linearly increases from 0 to 1 in the first half of training and remains at 1 in the second. Early on, γ_{en} is small, and the distances between v and the embedding vectors are almost the same, leading to a uniform $\boldsymbol{p}$. Therefore, all experts can be fully trained and gradually become specialized. In the later stages, $\mathcal{L}_{en}$ forces $\boldsymbol{p}$ to become one-hot based on specialized experts.

Due to the negative logarithm in Eq. (2b), the D2V encoder has to move towards one of the embedding vectors rather than away from the others in order to minimize $\mathcal{L}_{en}$. Therefore, the output of the D2V encoder will converge to $\mathcal{E}$ and become quantized during training.

3.4.3 Expert Load Balancing

Sparse MoE may suffer from an unbalanced expert load, which is problematic if only a small subset of experts are used while the others are left idle. To alleviate this problem, a widely used approach is to introduce an auxiliary importance loss $CV(I(X))^2$ [67], where X represents a single batch, $I(X) = [I_1(X), \cdots, I_K(X)]$ denotes the importance of experts, for which $I_k(X)$ is defined as the sum of gate values assigned to the kth expert (*i.e.*, sum the gate value matrix in Fig. 1b along the example dimension), and CV is the coefficient of variation. However, [58] showed that this importance loss over-penalizes unbalanced expert utilization and may be counter-productive, since in most cases the expert load is naturally unbalanced. In this case, [58] defined a distribution $P = I(X)/\sum I(X)$ and used the KL-divergence between P and the uniform distribution $\mathcal{U}$ to balance the expert load, which is also used in our work:

$$\mathcal{L}_{kl} = D_{KL}(P \| \mathcal{U}) = D_{KL}\left(\frac{I(X)}{\sum I(X)} \| \mathcal{U}\right) \quad (4)$$

Compared to the importance loss, $\mathcal{L}_{kl}$ achieves a better trade-off between expert specialization and load balancing.

3.5. Embedding Space

The embedding space $\mathcal{E}$ plays a key role in HMOE. As we can see, embedding vectors have an effect on both the generation of expert weights and the routing mechanism, thus serving as a bridge to balance these two parts. In addition, these embedding vectors are learnable like the weights and biases of neural networks and attract the D2V encoder during training under the influence of $\mathcal{L}_{en}$. This may be reminiscent of VQ-VAE [74], which also has an embedding space and makes its encoder output discrete latent codes.

3.6. Class-Adversarial Training on D2V

We would like to make the D2V encoder h_v less informative for classes, which ensures that HMOE partitions the input space based on domain-wise distinction rather than semantic categories. Inspired by Domain-Adversarial Neural Networks [22], we define an adversarial classifier f_c^{ad} taking v as input and add the following loss to perform class-adversarial training on h_v:

$$\mathcal{L}_{ad} = \mathbb{E}_{(x,y)\sim\mathcal{D}_{tr}}\left[\ell_{ce}(f_c^{ad}(GRL(h_v(x), \lambda_{grl})), y)\right] \quad (5)$$

where ℓ_{ce} denotes the cross-entropy loss and GRL represents the gradient reversal layer, which acts as an identity function in the forward pass and multiplies the gradient by

$-\lambda_{grl}$ in the backward pass. As suggested in [22], we define λ_{grl} as:

$$\lambda_{grl} = 2/(1 + \exp(-10 \times pct_{tr})) - 1 \tag{6}$$

where pct_{tr} denotes the training percentage varying linearly from 0 to 1.

3.7. Semi-/supervised Learning on Domains

Due to the probabilistic nature of MoE, given an input x and the corresponding gate values $\boldsymbol{p} = \{p_k\}_{k=1}^{K}$, we can interpret p_k as the probability of selecting the kth expert E_k given x, *i.e.*, $p(E_k|x)$. In addition, E_k is thought to be associated with a specific domain $\mathcal{S}_m$. Therefore, we get $p_k = p(E_k|x) = p(S_m|x)$. Consider a dataset with domain labels $\mathcal{D}_d = \{(x_i, d_i)\}_{i=1}^{N_d}$ (class labels are not necessary) with $d_i \in \{1, \cdots, M_d\}$, we can make use of $\mathcal{D}_d$ as follows:

$$\mathcal{L}_d = \mathbb{E}_{(x,d)\sim\mathcal{D}_d}\left[\ell_{ce}(\boldsymbol{p}, d)\right] \tag{7}$$

Note that M_d may be smaller than K, but this has no bearing on the calculation of $\mathcal{L}_d$. In this case, we assume that the first M_d experts are assigned to M_d domains, and the other experts do not have domain information and learn from the data by themselves. If all domain labels are available in the training data, $\mathcal{L}_d$ becomes supervised learning on domains.

3.8. Training and Inference

In addition to the above losses, the supervised loss on targets is:

$$\mathcal{L}_y = \mathbb{E}_{(x,y)\sim\mathcal{D}_{tr}}\left[\ell_{ce}(\hat{y}, y)\right] \tag{8}$$

where $\hat{y}$ is the prediction of HMOE, as calculated in Eq. (1). The final training loss is:

$$\mathcal{L} = \lambda_y\mathcal{L}_y + \lambda_{en}\mathcal{L}_{en} + \lambda_{kl}\mathcal{L}_{kl} + \lambda_{ad}\mathcal{L}_{ad} + \lambda_d\mathcal{L}_d \tag{9}$$

where λ are trade-off hyper-parameters to balance losses.

For inference, we provide three ways: MIX, MAX, and OOD. MIX means the mixture of experts, MAX uses the output of the expert with the highest gate value, and OOD[1] (Out of Domain) uses the output of a classifier whose weights are generated by the hypernetwork taking the D2V encoder as input.

4. Experiments

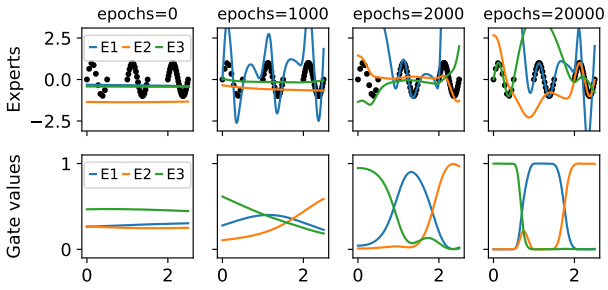
4.1. Toy Regression Problem

Although this work focuses on image classification, we start with a toy regression problem to gain some insight into the learning dynamics of HMOE, such as how gate values evolve and how experts become specialized gradually.

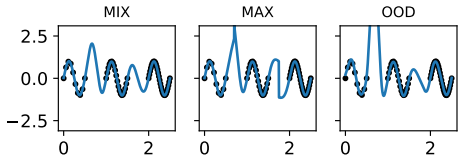
[1]The OOD inference can be efficiently implemented using PyTorch-based JAX-like library, *functorch*.

We use the function $y = \sin(4\pi x)$ to generate 10, 20, and 30 data points uniformly in three intervals: $[0, 0.5]$, $[1, 1.5]$ and $[2, 2.5]$, respectively. Unequal data points are used to simulate a naturally unbalanced expert load. All networks of HMOE are MLPs, and we create three embedding vectors of dimension $D = 8$. We employ $\mathcal{L}_y$ (use MSE as the loss function), $\mathcal{L}_{en}$, and $\mathcal{L}_{kl}$ with $\lambda_y = \lambda_{en} = \lambda_{kl} = 1$, and train HMOE using Adam [37] with learning rate 0.001 over $20,000$ epochs. More details are presented in the **supplementary material**.

The evolution of the experts' outputs and gates values w.r.t. training epochs is shown in Fig. 3a. We can see that three experts compete with each other and gradually locate their positions, and HMOE manages to identify three intervals even with imbalanced data. After training, we compare three modes of inference, as shown in Fig. 3b. They all coincide well with the training points. MIX seems to generalize best to the zones between intervals, while MAX has discontinuities due to hard switching between experts and OOD has an unexpected spike. Overall, HMOE demonstrates an ability to detect heterogeneous patterns in data.



(a) Experts' outputs and gate values during training



(b) Three modes of inference

Figure 3. A toy regression problem. We generate some data points using the function $y = \sin(4\pi x)$ in three intervals and fit HMOE with three embedding vectors to these points. HMOE well identifies three intervals and experts also become specialized.

4.2. DomainBed

4.2.1 Datasets and Model Evaluation

DomainBed [26] provides a unified codebase to implement and train DG algorithms and integrates some commonly used DG-related datasets. In this section, we experiment on Colored MNIST with 3 domains [2], Rotated MNIST with

6 domains [24], PACS with 4 domains [42], VLCS with 4 domains [17], OfficeHome with 4 domains [77], and TerraIncognita with 4 domains [4]. In the **supplementary material**, we give detailed statistics and visualize some samples for each domain of each dataset.

To select models and tune hyper-parameters, DomainBed gives three options, of which we choose the training-domain validation that randomly draws 80% from the data of each training domain to form the training set and uses the remaining as the validation set. This option best matches the setting of compound DG without domain labels and access to test domains.

4.2.2 Implementation Details

For Colored and Rotated MNIST, following [26], we use as the featurizer a four-layer ConvNet (refer to Appendix D.1 of [26]). The D2V encoder consists of two *conv* layers (32 units, 3×3 kernels, ReLU), followed by global average pooling and a fully-connected (fc) layer to map to the embedding dimension D.

For other datasets, we use ResNet50[2] pretrained on ImageNet [11] as the featurizer and freeze all batch normalization layers. The D2V encoder cascades 3 *conv* layers (64-128-256 units, stride 2, 4×4 kernels, ReLU), two residual blocks (each has 2 *conv* layers with 256 units, 3×3 kernels, ReLU), and a 3×3 *conv* layer with D units followed by global average pooling. In addition, we use Instance Normalization [73] with learnable affine parameters before all ReLU of the D2V encoder.

For all datasets, the classifier is a fc layer whose input size is the output size of the featurizer (128 for ConvNet and 2048 for ResNet50) and output size is the number of classes per dataset. The hypernetwork is a five-layer MLP with 256-128-64-32 hidden units and SiLU [30] except the output layer, and its input size is D and output size is the total number of learnable parameters (*i.e.*, weights and biases) of the classifier. If $\mathcal{L}_{ad}$ is used, the adversarial classifier is a three-layer MLP with 256 hidden units and ReLU except the output layer, and its input size is D and output size is the number of classes. In addition, we set $D = 32$ and initialize embedding vectors using the standard normal distribution.

We define three HMOE variants based on the number of embedding vectors K and whether domain labels are used: (1) **HMOE-DL:** Domain labels of $\mathcal{D}_{tr}$ are provided. In this case, we only use $\mathcal{L}_y$ and $\mathcal{L}_d$ with $\lambda_y = \lambda_d = 1$ and discard other losses, and K is the number of training domains per dataset. (2) **HMOE-DN:** Domain numbers are known but domain labels. In this case, K is the number of training domains per dataset. We use $\mathcal{L}_y$, $\mathcal{L}_{en}$, $\mathcal{L}_{kl}$, and $\mathcal{L}_{ad}$ with $\lambda_y = \lambda_{en} = \lambda_{kl} = 1$ and $\lambda_{ad} = 0.01$. (3) **HMOE-ND:** No domain information is available and we use a fixed $K = 5$. The setting of losses is the same as in HMOE-DN.

DomainBed trains all DG algorithms with Adam for 5,000 iterations. For Colored and Rotated MNIST / other datasets, the learning rate is 0.001 / 5e-5, the batch size is 64 / 32 $\times$ number of training domains, and models are evaluated on the validation set every 100 / 300 iterations. Each experiment uses one domain of a dataset as the test domain and trains algorithms on the others, which is repeated 3 times with different random seeds. The average accuracy over 3 replicates is reported. In addition, we do not tune hyper-parameters and use the settings mentioned above consistently. Other DG algorithms also use the default settings predefined in DomainBed. All experiments are performed on PyTorch using a A5000 GPU.

4.2.3 Results

We use the up-to-date domain generalization benchmark on DomainBed, and the comparison of our proposed HMOE (3 variants and 3 inference modes) with other DG algorithms is shown in Tab. 1, where the best results are underlined. ERM means the vanilla supervised learning that just finetunes ResNet50 on mixed domains, also called DeepAll in some papers and serving as a performance baseline. We list the average accuracy of all test domains for each dataset. Refer to the **supplementary material** for detailed results.

For Colored and Rotated MNIST, the performance of all algorithms is almost the same, except for the impressive results of ARM. Our proposed HMOE achieves SOTA results on PACS and TerraIncognita, which well demonstrates the effectiveness of HMOE. However, ERM outperforms HMOE and most DG algorithms for VLCS and OfficeHome. VLCS contains real camera photos, and its domain shift is mainly caused by changes in scene and perspective. We find that the visual differences between various domains of VLCS are subtle. In this case, forcing to reduce or model domain discrepancy may cause or aggravate overfitting. For OfficeHome, this is also the case.

Interestingly, HMOE-DL is inferior to HMOE-DN/ND in most cases, which implies that HMOE works better using its own learned domain information than using given domain labels. We find that latent domains discovered by HMOE are more human-intuitive than original domain labels (See Sec. 4.2.4).

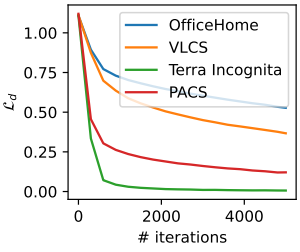


Figure 4. Avg. $\mathcal{L}_d$ per dataset

Fig. 4 shows the supervised loss on domains of HMOE-DL $\mathcal{L}_d$ w.r.t. iterations, which fails to decrease quickly for

[2]For a fair comparison with other DG algorithms, we use the pretrained ResNet50 of IMAGENET1K-V1 in PyTorch, although V2 is better.

Algorithm		ColoredMNIST	RotatedMNIST	VLCS	PACS	OfficeHome	TerraIncognita
w/ Domain Labels							
IRM [2]		52.0	97.7	78.5	83.5	64.3	47.6
GroupDRO [63]		52.1	98.0	76.7	84.4	66.0	43.2
Mixup [83]		52.1	98.0	77.4	84.6	68.1	47.9
MLDG [41]		51.5	97.9	77.2	84.9	66.8	47.7
CORAL [70]		51.5	98.0	**78.8**	86.2	**68.7**	47.6
MMD [44]		51.5	97.9	77.5	84.6	66.3	42.2
DANN [22]		51.5	97.8	78.6	83.6	65.9	46.7
CDANN [46]		51.7	97.9	77.5	82.6	65.8	45.8
MTL [5]		51.4	97.9	77.2	84.6	66.4	45.6
ARM [88]		**56.2**	**98.2**	77.6	85.1	64.8	45.5
VREx [38]		51.8	97.9	78.3	84.9	66.4	46.4
HMOE-DL	MIX	51.6	97.3	76.7	83.5	64.7	45.0
	MAX	51.7	97.0	77.6	83.9	63.2	43.2
	OOD	51.7	97.4	76.8	84.5	63.7	44.0
w/o Domain Labels							
ERM [76]		51.5	98.0	77.5	85.5	66.5	46.1
RSC [31]		51.7	97.6	77.1	85.2	65.5	46.6
SagNet [55]		51.7	98.0	77.8	86.3	68.1	48.6
HMOE-DN	MIX	51.9	97.5	76.8	84.8	65.4	48.7
	MAX	51.9	97.4	76.6	85.1	65.4	**49.5**
	OOD	51.9	97.5	75.8	84.9	65.3	48.4
HMOE-ND	MIX	51.6	97.5	76.6	84.5	65.5	48.4
	MAX	51.7	97.4	76.8	86.6	65.5	45.0
	OOD	51.7	97.5	76.7	**87.0**	65.6	47.1

Table 1. Domain generalization results on DomainBed

OfficeHome and VLCS. This means that the information of domain labels is not well absorbed and seems to be incompatible with HMOE. In addition, HMOE-DN / ND are basically tied in terms of performance. For the three modes of inference, MAX and OOD achieve competitive or better performance compared to MIX. Therefore, we can safely employ MAX and OOD in practice, which are more computationally efficient without computing all experts like MIX.

4.2.4 Latent Domain Discovery

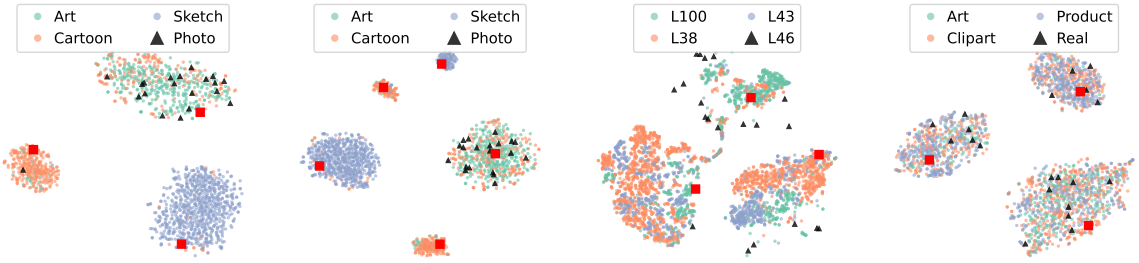
We use t-SNE [75] to visualize the output of the D2V encoder, as shown in Fig. 5. We can see that HMOE succeeds in partitioning the mixed data into a number of clusters, each around an embedding vector. The output of the D2V encoder converges to embedding vectors, which is as expected. For PACS (Fig. 5a), training domains are well separated. Some cartoon images look quite artistic and are classified as art. In addition, test photo samples are projected into the art cluster, which suggests that the D2V encoder should capture some semantics about latent domains since photo is closest to art. When we increases the number of embedding vectors K to 5, cartoon and sketch clusters are split into two sub-parts, as shown in Fig. 5b. For TerraIncognita (Fig. 5c), the dots of the same color are largely clustered together, and training domains are to some extent separated, although L38 and L43 are partially mixed. The test domain L46 seems to be more similar to L100. For OfficeHome (Fig. 5d), training domains are mixed in each cluster, which indicates a conflict between domain labels and inferred domains, and also explains why $\mathcal{L}_d$ cannot be reduced significantly for OfficeHome in Fig. 4.

To understand more intuitively how HMOE distinguishes between different domains, we visualize some samples to compare domain labels and HMOE clusters, as shown in Fig. 6. HMOE seemingly partitions TerraIncognita based on illumination and OfficeHome based on background complexity, which is more in line with human intuition about different domains than original domain labels.

After the above analysis, we conclude that the success of HMOE, *e.g.*, SOTA on PACS and TerraIncognita, is attributed to its ability to self-learn more reasonable and informative domain knowledge and use it efficiently.

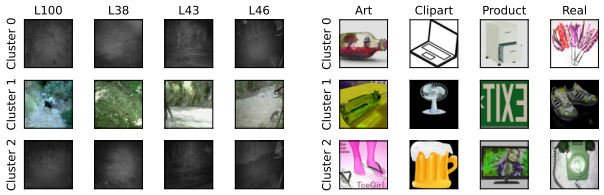
4.2.5 More Empirical Analysis

Ablation study We conduct an ablation study to analyze the contribution of class-adversarial learning. The results are shown in Tab. 2, which reports the average accuracy of three inference modes. As we can see, class-adversarial learning helps significantly improve performance in most cases, which validates its use and verifies the importance



(a) PACS-DN (b) PACS-ND ($K = 5$) (c) TerraIncognita-DN (d) OfficeHome-DN

Figure 5. The t-SNE visualization of the output of the D2V encoder. The suffixes in captions (DN and ND) represent HMOE-DN / ND, red squares are embedding vectors, black triangles are 20 samples randomly drawn from the test domain, and other dots are training domains.



(a) TerraIncognita-DN (b) OfficeHome-DN

Figure 6. Comparison between domain labels and HMOE clusters

and necessity of removing class-specific information from the D2V encoder.

	$\mathcal{L}_{ad}$	VLCS	PACS	OfficeHome	TerraIncognita
HMOE-DN	-	76.0	84.0	64.2	47.0
	✓	76.4	84.9	65.4	48.9
HMOE-ND	-	76.9	84.5	64.4	47.4
	✓	76.7	86.0	65.5	46.8

Table 2. Ablation study on class-adversarial learning

More embedding vectors We further increase K to 8, and we find that HMOE suffers from the learning collapse problem, *i.e.*, some embedding vectors collapse together and the D2V encoder outputs similar values, as shown in Fig. 7. When embedding vectors are much more than needed, HMOE encounters difficulties in how to assign the data to different experts and ends up with a large $\mathcal{L}_{kl}$. In this case, increasing λ_{kl} may alleviate the learning collapse.

Only supervised loss on targets Using only $\mathcal{L}_y$, we train HMOE with $K = 3$ on OfficeHome. In the absence of the entropy loss $\mathcal{L}_{en}$ forcing gate values to become one-hot, HMOE performs soft partitioning on the input space. The t-SNE visualization is shown in Fig. 8, which is obviously not comparable to Fig. 5d, for which clusters are distinctly separated. This demonstrates that the dense-to-sparse Top-1 routing algorithm works as expected and largely improves latent domain discovery compared to soft partitioning.

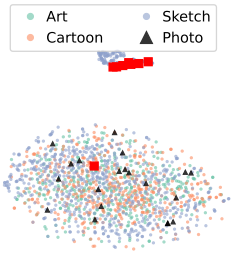


Figure 7. PACS-ND ($K = 8$) and learning collapse

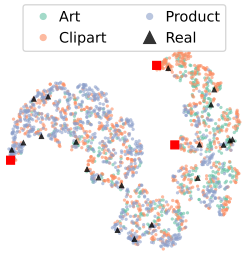


Figure 8. OfficeHome for HMOE ($K = 3$ and only $\mathcal{L}_y$)

5. Conclusion

This paper presents a novel method, HMOE, for compound DG without the need for domain labels. Compared to other methods requiring domain labels, HMOE shows competitive performance and achieves SOTA results on PACS and TerraIncognita datasets. In addition, HMOE exhibits the distinctive property of latent domain discovery. It is worth mentioning that the discovery and utilization of domain information are jointly undertaken rather than in stages like other related work. The key to our work is to use Mixture of Experts (MoE) and leverage its *divide and conquer* ability. In addition, we leverage hypernetworks to generate the weights of expert networks.

However, it remains unclear how to effectively determine an appropriate number of experts or embedding vectors to fully explore domain information while avoiding the learning collapse. A promising but challenging solution that we will explore in future work is to use tree-structured hierarchical MoE to discover hierarchical domain knowledge, where each level contains only a number of experts but the number of multi-level inferred domains grows exponentially. Moreover, our proposed HMOE is versatile and scalable, and it should also be applicable to a wide range of problems beyond the scope of DG that are troubled by heterogeneous patterns.

References

[1] Kartik Ahuja, Ethan Caballero, Dinghuai Zhang, Jean-Christophe Gagnon-Audet, Yoshua Bengio, Ioannis Mitliagkas, and Irina Rish. Invariance principle meets information bottleneck for out-of-distribution generalization. *Advances in Neural Information Processing Systems*, 34:3438–3450, 2021. 2

[2] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019. 2, 5, 7, 1, 3, 4

[3] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. *Advances in neural information processing systems*, 31, 2018. 2

[4] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 456–473, 2018. 6, 1

[5] Gilles Blanchard, Aniket Anand Deshmukh, Ürun Dogan, Gyemin Lee, and Clayton Scott. Domain generalization by marginal transfer learning. *The Journal of Machine Learning Research*, 22(1):46–100, 2021. 1, 2, 7, 3, 4

[6] Dhanajit Brahma, Vinay Kumar Verma, and Piyush Rai. Hypernetworks for Continual Semi-Supervised Learning. *arXiv preprint arXiv:2110.01856*, 2021. 2

[7] Fabio M. Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2019. 2

[8] Oscar Chang, Lampros Flokas, and Hod Lipson. Principled weight initialization for hypernetworks. In *International Conference on Learning Representations*, 2019. 3

[9] Chaoqi Chen, Jiongcheng Li, Xiaoguang Han, Xiaoqing Liu, and Yizhou Yu. Compound Domain Generalization via Meta-Knowledge Encoding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7119–7129, 2022. 1, 2

[10] Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. StableMoE: Stable routing strategy for mixture of experts. *arXiv preprint arXiv:2204.08396*, 2022. 3

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[12] Aniket Anand Deshmukh, Yunwen Lei, Srinagesh Sharma, Urun Dogan, James W. Cutler, and Clayton Scott. A generalization error bound for multi-class domain generalization. *arXiv preprint arXiv:1905.10392*, 2019. 1

[13] Antonio D'Innocente and Barbara Caputo. Domain generalization with domain-specific aggregation modules. In *German Conference on Pattern Recognition*, pages 187–198. Springer, 2018. 2

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, and Sylvain Gelly. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[15] Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[16] Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, and Orhan Firat. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022. 2

[17] Chen Fang, Ye Xu, and Daniel N. Rockmore. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1657–1664, 2013. 6, 1

[18] William Fedus, Jeff Dean, and Barret Zoph. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*, 2022. 2

[19] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. 2

[20] Chuang Gan, Tianbao Yang, and Boqing Gong. Learning attributes equals multi-source domain generalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 87–97, 2016. 2

[21] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189. PMLR, 2015. 2

[22] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016. 2, 4, 5, 7, 3

[23] Muhammad Ghifary, David Balduzzi, W. Bastiaan Kleijn, and Mengjie Zhang. Scatter component analysis: A unified framework for domain adaptation and domain generalization. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1414–1430, 2016. 2

[24] Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2551–2559, 2015. 2, 6, 1

[25] Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool. Dlow: Domain flow for adaptation and generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2477–2486, 2019. 2

[26] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020. 1, 2, 3, 5, 6

[27] David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 1, 2

[28] Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder,

Lichan Hong, and Ed Chi. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *Advances in Neural Information Processing Systems*, 34:29335–29347, 2021. 3

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 3

[30] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 6, 1

[31] Zeyi Huang, Haohan Wang, Eric P. Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *European Conference on Computer Vision*, pages 124–140. Springer, 2020. 2, 7, 3, 4

[32] Maximilian Ilse, Jakub M Tomczak, Christos Louizos, and Max Welling. Diva: Domain invariant variational autoencoders. In *Medical Imaging with Deep Learning*, pages 322–348. PMLR, 2020. 2

[33] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991. 1, 2

[34] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, 6(2):181–214, 1994. 1, 2

[35] Aditya Khosla, Tinghui Zhou, Tomasz Malisiewicz, Alexei A. Efros, and Antonio Torralba. Undoing the damage of dataset bias. In *European Conference on Computer Vision*, pages 158–171. Springer, 2012. 2

[36] Daehee Kim, Youngjun Yoo, Seunghyun Park, Jinkyu Kim, and Jaekoo Lee. Selfreg: Self-supervised contrastive regularization for domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9619–9628, 2021. 2

[37] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[38] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pages 5815–5826. PMLR, 2021. 2, 7, 3, 4

[39] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020. 2

[40] Bo Li, Jingkang Yang, Jiawei Ren, Yezhen Wang, and Ziwei Liu. Sparse Fusion Mixture-of-Experts are Domain Generalizable Learners. *arXiv preprint arXiv:2206.04046*, 2022. 2

[41] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 2, 7, 3, 4

[42] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5542–5550, 2017. 6, 1

[43] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M. Hospedales. Episodic training for domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1446–1455, 2019. 2

[44] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C. Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018. 2, 7, 3, 4

[45] Pan Li, Da Li, Wei Li, Shaogang Gong, Yanwei Fu, and Timothy M. Hospedales. A simple feature augmentation for domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8886–8895, 2021. 2

[46] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 624–639, 2018. 2, 7, 3, 4

[47] Xi Lin, Zhiyuan Yang, Qingfu Zhang, and Sam Kwong. Controllable pareto multi-task learning. *arXiv preprint arXiv:2010.06313*, 2020. 2

[48] Alexander H. Liu, Yen-Cheng Liu, Yu-Ying Yeh, and Yu-Chiang Frank Wang. A unified feature disentangler for multi-domain image translation and manipulation. *Advances in neural information processing systems*, 31, 2018. 2

[49] Jonathan Lorraine and David Duvenaud. Stochastic hyperparameter optimization through hypernetworks. *arXiv preprint arXiv:1802.09419*, 2018. 2

[50] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. *arXiv preprint arXiv:2106.04489*, 2021. 2

[51] Massimiliano Mancini, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. Best sources forward: Domain generalization through source-specific nets. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 1353–1357. IEEE, 2018. 2

[52] Toshihiko Matsuura and Tatsuya Harada. Domain generalization using a mixture of multiple latent domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11749–11756, 2020. 2

[53] Saeid Motiian, Marco Piccirilli, Donald A. Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5715–5725, 2017. 2

[54] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18. PMLR, 2013. 1, 2

[55] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap by reducing style bias. In *Proceedings of the IEEE/CVF Conference*

on Computer Vision and Pattern Recognition, pages 8690–8699, 2021. 2, 7, 3, 4

[56] Aviv Navon, Aviv Shamsian, Gal Chechik, and Ethan Fetaya. Learning the pareto front with hypernetworks. *arXiv preprint arXiv:2010.04104*, 2020. 2

[57] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE transactions on neural networks*, 22(2):199–210, 2010. 2

[58] Svetlana Pavlitskaya, Christian Hubschneider, Lukas Struppek, and J. Marius Zöllner. Balancing Expert Utilization in Mixture-of-Experts Layers Embedded in CNNs. *arXiv preprint arXiv:2204.10598*, 2022. 4

[59] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1406–1415, 2019. 2

[60] Xingchao Peng, Zijun Huang, Ximeng Sun, and Kate Saenko. Domain agnostic learning with disentangled representations. In *International Conference on Machine Learning*, pages 5102–5112. PMLR, 2019. 2

[61] Fengchun Qiao, Long Zhao, and Xi Peng. Learning to learn single domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12556–12565, 2020. 2

[62] Alexandre Rame, Corentin Dancette, and Matthieu Cord. Fishr: Invariant gradient variances for out-of-distribution generalization. In *International Conference on Machine Learning*, pages 18347–18377. PMLR, 2022. 2

[63] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization, Apr. 2020. 2, 7, 3, 4

[64] Marcin Sendera, Marcin Przewięźlikowski, Konrad Karanowski, Maciej Zięba, Jacek Tabor, and Przemysław Spurek. Hypershot: Few-shot learning by kernel hypernetworks. *arXiv preprint arXiv:2203.11378*, 2022. 2

[65] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In *International Conference on Machine Learning*, pages 9489–9502. PMLR, 2021. 2

[66] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745*, 2018. 2

[67] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. 2, 4

[68] Yuge Shi, Jeffrey Seely, Philip HS Torr, N. Siddharth, Awni Hannun, Nicolas Usunier, and Gabriel Synnaeve. Gradient matching for domain generalization. *arXiv preprint arXiv:2104.09937*, 2021. 2

[69] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[70] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*, pages 443–450. Springer, 2016. 2, 7, 3, 4

[71] Yi Tay, Zhe Zhao, Dara Bahri, Don Metzler, and Da-Cheng Juan. Hypergrid transformers: Towards a single model for multiple tasks. In *ICLR 2021*, 2021. 2

[72] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 2

[73] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 6

[74] Aaron Van Den Oord and Oriol Vinyals. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 4

[75] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 7

[76] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999. 7, 2, 3, 4

[77] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5018–5027, 2017. 6, 1

[78] Tomer Volk, Eyal Ben-David, Ohad Amosy, Gal Chechik, and Roi Reichart. Example-based hypernetworks for out-of-distribution generalization. *arXiv preprint arXiv:2203.14276*, 2022. 2

[79] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C. Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. *Advances in neural information processing systems*, 31, 2018. 2

[80] Johannes Von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019. 2

[81] Jindong Wang, Wenjie Feng, Yiqiang Chen, Han Yu, Meiyu Huang, and Philip S. Yu. Visual domain adaptation with manifold embedded distribution alignment. In *Proceedings of the 26th ACM International Conference on Multimedia*, pages 402–410, 2018. 2

[82] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 2022. 1, 3

[83] Shen Yan, Huan Song, Nanxiang Li, Lincan Zou, and Liu Ren. Improve unsupervised domain adaptation with mixup training. *arXiv preprint arXiv:2001.00677*, 2020. 7, 2, 3, 4

[84] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2100–2110, 2019. 2

[85] Seniha Esen Yuksel, Joseph N. Wilson, and Paul D. Gader. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193, 2012. 1

[86] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2

[87] Hanlin Zhang, Yi-Fan Zhang, Weiyang Liu, Adrian Weller, Bernhard Schölkopf, and Eric P Xing. Towards principled disentanglement for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8024–8034, 2022. 2

[88] Marvin Zhang, Henrik Marklund, Nikita Dhawan, Abhishek Gupta, Sergey Levine, and Chelsea Finn. Adaptive risk minimization: Learning to adapt to domain shift. *Advances in Neural Information Processing Systems*, 34:23664–23678, 2021. 7, 2, 3, 4

[89] Dominic Zhao, Johannes von Oswald, Seijin Kobayashi, João Sacramento, and Benjamin F Grewe. Meta-learning via hypernetworks. In *4th Workshop on Meta-Learning at NeurIPS 2020 (MetaLearn 2020)*, 2020. 2

[90] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1, 3

[91] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Learning to generate novel domains for domain generalization. In *European Conference on Computer Vision*, pages 561–578. Springer, 2020. 2

[92] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain adaptive ensemble learning. *IEEE Transactions on Image Processing*, 30:8008–8018, 2021. 2

[93] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. *arXiv preprint arXiv:2104.02008*, 2021. 2

[94] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. Designing effective sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022. 2

HMOE: Hypernetwork-based Mixture of Experts for Domain Generalization

Supplementary Material

A. Toy Regression Problem

For the toy regression problem, HMOE uses three embedding vectors of dimension $D = 8$, which are initialized using the standard normal distribution. All networks of HMOE are MLPs. The featurizer is a three-layer MLP with 32 hidden units, and its input size is 1 and output size is 32. The encoder is a three-layer MLP with 32 hidden units, and its input size is 1 and output size is D. The classifier is a two-layer MLP with 32 hidden units, and its input size is 32 and output size is 1. The hypernetwork is a four-layer MLP with 32 hidden units, and its input size is D and output size is the total number of learnable parameters (*i.e.*, weights and biases) of the classifier. In addition, all MLPs use SiLU [30] except the output layers.

B. DomainBed

B.1. Description and visualization of datasets

Dataset	Domains	# of classes	# of samples	Image size
ColoredMNIST [2]	+90%  +80%  -90%  *(degree of correlation between color and label)*	2	70,000	(2, 28, 28)
RotatedMNIST [24]	0°  15°  30°  45°  60°  75° 	10	70,000	(1, 28, 28)
VLCS [17]	Caltech101  LabelMe  SUN09  VOC2007 	5	10,729	(3, 224, 224)
PACS [42]	Art  Cartoon  Photo  Sketch 	7	9,991	(3, 224, 224)
OfficeHome [77]	Art  Clipart  Product  Photo 	65	15,588	(3, 224, 224)
TerraIncognita [4]	L100  L38  L43  L46 	10	24,788	(3, 224, 224)
	(camera trap location)			

Table 3. Description and visualization of datasets used in our experiments (Adapted from [26])

B.2. Detailed domain generalization results

We provide the domain generalization results of each test domain for each dataset, and the best results are underlined.

Algorithm		+90%	+80%	-90%	Avg
w/ Domain Labels					
IRM [2]		72.5	73.3	10.2	52.0
GroupDRO [63]		73.1	73.2	10.0	52.1
Mixup [83]		72.7	73.4	10.1	52.1
MLDG [41]		71.5	73.1	9.8	51.5
CORAL [70]		71.6	73.1	9.9	51.5
MMD [44]		71.4	73.1	9.9	51.5
DANN [22]		71.4	73.1	10.0	51.5
CDANN [46]		72.0	73.0	10.2	51.7
MTL [5]		70.9	72.8	10.5	51.4
ARM [88]		**82.0**	**76.5**	10.2	**56.2**
VREx [38]		72.4	72.9	10.2	51.8
HMOE-DL	MIX	71.8	72.9	10.1	51.6
	MAX	71.9	73.1	10.1	51.7
	OOD	71.9	73.2	10.1	51.7
w/o Domain Labels					
ERM [76]		71.7	72.9	10.0	51.5
RSC [31]		71.9	73.1	10.0	51.7
SagNet [55]		71.8	73.0	10.3	51.7
HMOE-DN	MIX	72.0	73.0	**10.7**	51.9
	MAX	72.0	73.0	**10.7**	51.9
	OOD	72.0	73.0	**10.7**	51.9
HMOE-ND	MIX	71.6	73.2	10.1	51.6
	MAX	71.9	73.2	10.2	51.7
	OOD	71.6	73.5	10.1	51.7

Table 4. Domain generalization results on ColoredMNIST

Algorithm		0	15	30	45	60	75	Avg
w/ Domain Labels								
IRM [2]		95.5	98.8	98.7	98.6	98.7	95.9	97.7
GroupDRO [63]		95.6	98.9	98.9	99.0	98.9	**96.5**	98.0
Mixup [83]		95.8	98.9	98.9	98.9	98.8	**96.5**	98.0
MLDG [41]		95.8	98.9	**99.0**	98.9	99.0	95.8	97.9
CORAL [70]		95.8	98.8	98.9	99.0	98.9	96.4	98.0
MMD [44]		95.6	98.9	**99.0**	99.0	98.9	96.0	97.9
DANN [22]		95.0	98.9	**99.0**	99.0	98.9	96.3	97.8
CDANN [46]		95.7	98.8	98.9	98.9	98.9	96.1	97.9
MTL [5]		95.6	99.0	**99.0**	98.9	99.0	95.8	97.9
ARM [88]		**96.7**	**99.1**	**99.0**	99.0	**99.1**	**96.5**	**98.2**
VREx [38]		95.9	99.0	98.9	98.9	98.7	96.2	97.9
HMOE-DL	MIX	94.5	97.7	98.8	98.7	99.0	94.9	97.3
	MAX	94.1	97.9	98.5	98.5	98.6	94.7	97.0
	OOD	95.0	97.9	98.6	98.9	**99.1**	94.7	97.4
w/o Domain Labels								
ERM [76]		95.9	98.9	98.8	98.9	98.9	96.4	98.0
RSC [31]		94.8	98.7	98.8	98.8	98.9	95.9	97.6
SagNet [55]		95.9	98.9	**99.0**	**99.1**	99.0	96.3	98.0
HMOE-DN	MIX	94.1	98.6	98.7	98.6	99.0	95.9	97.5
	MAX	94.0	98.6	98.7	98.6	98.8	95.9	97.4
	OOD	94.0	98.6	98.7	98.6	99.0	95.9	97.5
HMOE-ND	MIX	94.1	98.6	98.7	98.6	99.0	95.9	97.5
	MAX	94.0	98.6	98.7	98.6	98.8	95.9	97.4
	OOD	94.0	98.6	98.7	98.6	99.0	95.9	97.5

Table 5. Domain generalization results on RotatedMNIST

Algorithm		Caltech101	LabelMe	SUN09	VOC2007	Avg
w/ Domain Labels						
IRM [2]		98.6	64.9	73.4	77.3	78.5
GroupDRO [63]		97.3	63.4	69.5	76.7	76.7
Mixup [83]		98.3	64.8	72.1	74.3	77.4
MLDG [41]		97.4	65.2	71.0	75.3	77.2
CORAL [70]		98.3	**66.1**	73.4	**77.5**	**78.8**
MMD [44]		97.7	64.0	72.8	75.3	77.5
DANN [22]		**99.0**	65.1	73.1	77.2	78.6
CDANN [46]		97.1	65.1	70.7	77.1	77.5
MTL [5]		97.8	64.3	71.5	75.3	77.2
ARM [88]		98.7	63.6	71.3	76.7	77.6
VREx [38]		98.4	64.4	**74.1**	76.2	78.3
HMOE-DL	MIX	97.7	62.3	72.0	74.8	76.7
	MAX	97.0	63.6	73.2	76.7	77.6
	OOD	97.0	63.4	72.0	74.9	76.8
w/o Domain Labels						
ERM [76]		97.7	64.3	73.4	74.6	77.5
RSC [31]		97.9	62.5	72.3	75.6	77.1
SagNet [55]		97.9	64.5	71.4	**77.5**	77.8
HMOE-DN	MIX	97.1	63.8	71.2	75.1	76.8
	MAX	97.4	63.4	70.9	74.8	76.6
	OOD	96.4	62.9	69.3	74.7	75.8
HMOE-ND	MIX	95.8	65.7	72.4	72.5	76.6
	MAX	97.3	61.7	72.1	76.1	76.8
	OOD	96.9	61.8	72.0	76.0	76.7

Table 6. Domain generalization results on VLCS

Algorithm		Art	Cartoon	Photo	Sketch	Avg
w/ Domain Labels						
IRM [2]		84.8	76.4	96.7	76.1	83.5
GroupDRO [63]		83.5	79.1	96.7	78.3	84.4
Mixup [83]		86.1	78.9	**97.6**	75.8	84.6
MLDG [41]		85.5	80.1	97.4	76.6	84.9
CORAL [70]		88.3	80.0	97.5	78.8	86.2
MMD [44]		86.1	79.4	96.6	76.5	84.6
DANN [22]		86.4	77.4	97.3	73.5	83.6
CDANN [46]		84.6	75.5	96.8	73.5	82.6
MTL [5]		87.5	77.1	96.4	77.3	84.6
ARM [88]		86.8	76.8	97.4	79.3	85.1
VREx [38]		86.0	79.1	96.9	77.7	84.9
HMOE-DL	MIX	84.1	77.3	96.3	76.4	83.5
	MAX	82.9	78.6	95.9	78.1	83.9
	OOD	85.0	78.3	95.3	79.4	84.5
w/o Domain Labels						
ERM [76]		84.7	80.8	97.2	79.3	85.5
RSC [31]		85.4	79.7	**97.6**	78.2	85.2
SagNet [55]		87.4	80.7	97.1	80.0	86.3
HMOE-DN	MIX	83.9	82.3	95.0	77.9	84.8
	MAX	84.7	82.4	96.4	76.9	85.1
	OOD	83.9	80.2	95.6	**80.1**	84.9
HMOE-ND	MIX	88.8	78.9	96.6	73.9	84.5
	MAX	88.8	82.7	95.7	79.1	86.6
	OOD	**88.9**	**84.3**	95.7	79.0	**87.0**

Table 7. Domain generalization results on PACS

Algorithm		Art	Clipart	Product	Real	Avg
w/ Domain Labels						
IRM [2]		58.9	52.2	72.1	74.0	64.3
GroupDRO [63]		60.4	52.7	75.0	76.0	66.0
Mixup [83]		62.4	**54.8**	**76.9**	78.3	68.1
MLDG [41]		61.5	53.2	75.0	77.5	66.8
CORAL [70]		**65.3**	54.4	76.5	**78.4**	**68.7**
MMD [44]		60.4	53.3	74.3	77.4	66.3
DANN [22]		59.9	53.0	73.6	76.9	65.9
CDANN [46]		61.5	50.4	74.4	76.6	65.8
MTL [5]		61.5	52.4	74.9	76.8	66.4
ARM [88]		58.9	51.0	74.1	75.2	64.8
VREx [38]		60.7	53.0	75.3	76.6	66.4
HMOE-DL	MIX	59.5	50.5	73.6	75.2	64.7
	MAX	58.5	47.7	72.5	74.1	63.2
	OOD	58.6	49.9	72.8	73.7	63.7
w/o Domain Labels						
ERM [76]		61.3	52.4	75.8	76.6	66.5
RSC [31]		60.7	51.4	74.8	75.1	65.5
SagNet [55]		63.4	**54.8**	75.8	78.3	68.1
HMOE-DN	MIX	59.4	52.9	74.6	74.7	65.4
	MAX	60.0	52.1	74.6	74.9	65.4
	OOD	60.2	52.5	73.6	74.7	65.3
HMOE-ND	MIX	60.0	52.4	74.3	75.1	65.5
	MAX	60.0	52.4	73.3	76.3	65.5
	OOD	60.0	54.1	72.8	75.6	65.6

Table 8. Domain generalization results on OfficeHome

Algorithm		L100	L38	L43	L46	Avg
w/ Domain Labels						
IRM [2]		54.6	39.8	56.2	39.6	47.6
GroupDRO [63]		41.2	38.6	56.7	36.4	43.2
Mixup [83]		**59.6**	42.2	55.9	33.9	47.9
MLDG [41]		54.2	44.3	55.6	36.9	47.7
CORAL [70]		51.6	42.2	57.0	39.8	47.6
MMD [44]		41.9	34.8	57.0	35.2	42.2
DANN [22]		51.1	40.6	57.4	37.7	46.7
CDANN [46]		47.0	41.3	54.9	39.8	45.8
MTL [5]		49.3	39.6	55.6	37.8	45.6
ARM [88]		49.3	38.3	55.8	38.7	45.5
VREx [38]		48.2	41.7	56.8	38.7	46.4
HMOE-DL	MIX	43.1	44.9	55.8	36.1	45.0
	MAX	42.2	38.1	55.0	37.4	43.2
	OOD	43.0	42.2	54.6	36.3	44.0
w/o Domain Labels						
ERM [76]		49.8	42.1	56.9	35.7	46.1
RSC [31]		50.2	39.2	56.3	**40.8**	46.6
SagNet [55]		53.0	43.0	57.9	40.4	48.6
HMOE-DN	MIX	54.0	43.6	**58.3**	38.8	48.7
	MAX	54.0	**47.0**	**58.3**	38.8	**49.5**
	OOD	54.1	42.1	**58.3**	39.0	48.4
HMOE-ND	MIX	58.8	41.8	56.1	36.8	48.4
	MAX	44.7	41.8	56.5	36.8	45.0
	OOD	53.4	41.8	55.6	37.5	47.1

Table 9. Domain generalization results on TerraIncognita